A Project Report on

BSCY GRADE CALCULATOR FOR STUDENT WITH COURSE-BASED CRITERIA

Grade Calculator For Student With Course-Based Criteria



by

**Mustafa Khan BCY243029**

**Saif-Ul-Islam BCY243030**

A Project Report submitted to the DEPARTMENT OF ELECTRICAL AND COMPUTER

ENGINEERING

in partial fulfillment of the requirements for the degree of BACHELORS OF SCIENCE IN CYBER SECURITY

Faculty of Engineering

Capital University of Science & Technology, Islamabad

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Content

Problems………………………………………………………3-4

Methodologies ………………………………………………..5-7

Result of the Project …………………………………………..8

Reference of The Project………………………………………9-10

Code of The Project…………………………………………10-17

**Problem:** The goal is to develop a grade calculator that evaluates student performance based on predefined course criteria. The system should calculate a student’s final grade by considering multiple assessment components, each with different weights or percentage contributions. These components can include assignments, exams, quizzes, participation, projects, and any other evaluative measures set by the course instructor.

**Description of the Problem:** In many academic settings, students’ final grades are based on a combination of multiple assessments. The final grade is calculated by assigning different weights to various assessment categories, each reflecting its importance in the overall grading scheme. However, manually calculating final grades can lead to errors, inconsistency, and confusion for both students and instructors.

For instance, an instructor may decide that:

* Assignments contribute 30% to the final grade,
* Midterm exam contributes 40%,
* Final exam contributes 30%.

If students or instructors attempt to calculate the final grades manually, mistakes can occur, and it can become difficult to keep track of performance over time, especially when dealing with large numbers of students.

**Core Requirements for the Grade Calculator:**

1. **Input Data**:

* The system should allow the input of grades for various components (assignments, exams, quizzes, etc.).
* The system should allow the input of the corresponding weight or percentage for each component.
* The grades should be normalized (e.g., out of 100, GPA scale, etc.) depending on the grading structure used by the course.

1. **Calculation of Final Grade**:

* The system must automatically calculate the final grade based on the grades entered for each component and the weight assigned to each component.
* The final grade must be represented as a percentage or letter grade (depending on the system used).
* The calculation should be accurate, and the results must be displayed clearly to the student and instructor.

1. **Grade Evaluation**:
   * The system must account for different grading scales, such as:
     + Numeric scale (0-100),
     + Letter grade scale (A, B, C, D, F).
   * The system should also allow for the option to display either the final percentage or the letter grade, or both.
2. **Error Handling**:

* The system should handle edge cases, such as invalid grade entries (e.g., a grade greater than the maximum possible score) or inconsistent input where the weights do not add up to 100%.
* The system should notify users of any errors with clear messages.

1. **User Interface**:

* The system should provide an easy-to-use interface for students and instructors.
* It should allow users to enter multiple grade categories, view their input, and instantly calculate the result.
* The user interface should be accessible and visually clear, with a clear distinction between inputs (grades and weights) and outputs (final grade, error messages).

**Challenges in Implementing the System:**

* Different courses may have different grading schemes, and the system should be flexible enough to handle a variety of input types.
* Accuracy in calculating grades based on the weight of each assessment category is critical.
* Ensuring that the system handles edge cases such as invalid inputs and missing data will be essential to prevent erroneous grade calculations.

**Methodologies:** The methodologies should be clearly outlined to show the steps and approaches taken during the development of the project. Below are the key methodologies used in the design and implementation of the grade calculator:

### 1. **Requirement Analysis**

* **Purpose**: To understand the needs of the users (students and instructors) and define the necessary course criteria for grading.
* **Methodology**:
  + Conduct a discussion with stakeholders (e.g., instructors, students) to understand their grading criteria (e.g., assignments, quizzes, midterm, final exams).
  + Identify the required input and output for the program. This includes deciding on what data needs to be entered (grades and weights) and what the output should be (final grade and letter grade).
  + Research common grading schemes used in educational institutions to ensure flexibility in the calculator.

### 2. **System Design**

* **Purpose**: To create a clear structure for the grade calculator program, ensuring that it meets the requirements identified.
* **Methodology**:
  + **Modular Design**: Divide the system into smaller functional modules, such as input handling, grade calculation, and result display.
    - Functions for inputting grades and weights.
    - A function for calculating the final grade.
    - A function for displaying the results.
  + **Flowchart**: Design a flowchart that illustrates the flow of data through the system (from input to output).

### 3. **Algorithm Design**

* **Purpose**: To create the underlying logic that will calculate the final grade based on weighted inputs.
* **Methodology**:
  + **Weighted Average Calculation**: Develop an algorithm to compute the weighted average of grades. For example, multiplying each grade by its corresponding weight and summing up the results to calculate the final grade:

final\_grade = (grade1 \* weight1 + grade2 \* weight2 + grade3 \* weight3 + grade4 \* weight4) / 100

* **Conditional Logic for Letter Grade**: Create conditional statements to assign letter grades based on the final numerical grade:
* For instance, if the grade is >= 90, assign an "A", if it’s between 80 and 89, assign a "B", and so on

### 4. **Programming Implementation**

* **Purpose**: To translate the design and algorithms into code using C++.
* **Methodology**:
  + **Input Handling**: Use **cin** to collect grades and weights from the user. Ensure input validation for valid grade and weight entries.
  + **Grade Calculation**: Implement the weighted average calculation inside a function.
  + **Output Display**: Use **cout** to display the calculated final grade and the corresponding letter grade.
  + **Error Handling**: Although this is a simple program, handling basic errors such as invalid inputs (e.g., grades out of range, weights not adding up to 100%) is important.
  + **Function Usage**: Organize the code into functions for each major task (input, calculation, and output) to make the code modular and reusable.

### 5. **Testing and Debugging**

* **Purpose**: To ensure that the program works as expected and produces correct results for various inputs.
* **Methodology**:
  + **Unit Testing**: Test individual components of the program (e.g., grade input, final grade calculation).
  + **Test Cases**: Use a variety of test cases with different inputs:
    - Valid inputs for all grades and weights.
    - Edge cases such as grades at the boundaries of the grading scale (e.g., 0%, 100%).
    - Ensure weights are correctly summed to 100%.
  + **Manual and Automated Testing**: While this is a simple program, manual testing of various inputs can be done to ensure correctness.
  + **Debugging**: Use debugging tools in an IDE (like Visual Studio or Code Blocks) to identify and fix any errors that arise during testing.

### 6. **User Interface (UI) Design**

* **Purpose**: To ensure that the program is easy to use for students or instructors who input data.
* **Methodology**:
  + **Console-Based UI**: Since the program is a simple console application, design a clear text-based interface for interacting with the user. Ensure that prompts are easy to understand (e.g., "Enter grade for Assignment" and "Enter weight for Assignment").
  + **Clarity and Simplicity**: Provide clear instructions on how to input grades and weights, and display results in a readable format.

### 8. **Performance Evaluation**

* **Purpose**: To ensure that the program runs efficiently and handles expected use cases without performance issues.
* **Methodology**:
  + **Efficiency Considerations**: Since the grade calculator is a small application with minimal computation, performance is not a major concern. However, it’s important to ensure the program handles a reasonable number of inputs (e.g., if extended to multiple students).
  + **Stress Testing**: Ensure the program behaves correctly even with unexpected or extreme input values (e.g., extremely high or low grades, missing input).

### 9. **Feedback and Continuous Improvement**

* **Purpose**: To continuously improve the program based on feedback from users or stakeholders.
* **Methodology**:
  + **User Feedback**: After the program is used by students and instructors, gather feedback on the usability and functionality of the grade calculator.
  + **Feature Enhancements**: Based on feedback, consider adding new features such as:
    - Ability to handle more grading components.
    - Support for different grading schemes (e.g., GPA scale).
    - Error handling for invalid inputs (e.g., non-numeric entries).

**Result of The Project**

### **Example Scenario:**

* **Assignments**: Grade = 85, Weight = 20%
* **Quizzes**: Grade = 75, Weight = 20%
* **Midterm Exam**: Grade = 80, Weight = 20%
* **Final Exam**: Grade = 90, Weight = 40%

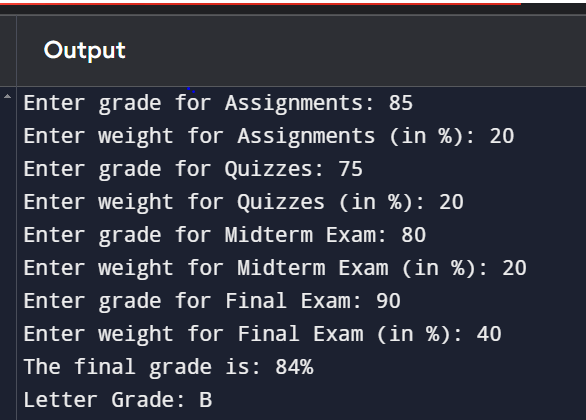
### **Step-by-Step Calculation:**

1. **Input Grades and Weights**:
   1. **Assignment**: 85 (Grade), 20% (Weight)
   2. **Quiz**: 75 (Grade), 20% (Weight)
   3. **Midterm Exam**: 80 (Grade), 20% (Weight)
   4. **Final Exam**: 90 (Grade), 40% (Weight)
2. **Formula to calculate grade:**
   * 1. final\_grade = (grade1 \* weight1 + grade2 \* weight2 + grade3 \* weight3 + grade4 \* weight4) / 100
3. **Final Grade**: The final calculated grade is **84.0%**
4. **Letter Grade**: Based on the grading scale defined in the program:

* **A**: 90% and above
* **B**: 80% to 89%
* **C**: 70% to 79%
* **D**: 60% to 69%
* **F**: Below 60%

Since the final grade is **83.0%**, the letter grade would be **B**.

**Output of the Project (Grade Calculator):**



**References of The Project**

### 1. **Textbooks and Academic Resources**:

* **Textbooks on C++ Programming**: To understand the syntax, functions, and object-oriented principles in C++, you might refer to textbooks that cover fundamental C++ programming concepts.
  + Example:
    - **"C++ Primer" by Stanley B. Lippman**
    - **"Accelerated C++" by Andrew Koenig**
* **Educational Resources on Grading Systems**: Many educational institutions use weighted averages to calculate grades. Resources on grading systems might provide insights into the weights and how grades are typically calculated.

### 2. **C++ Documentation**:

* Official documentation for C++ provides details on the syntax, standard libraries, functions, and best practices that may be used in the development of the program.
  + **C++ Reference**: cppreference.com
  + **C++ Standard Library Documentation**: This can provide information on input/output operations like cin and cout.

### 3. **Online Programming Tutorials**:

* To get guidance on specific C++ concepts (e.g., functions, loops, conditional statements), you might refer to online tutorials or websites.
  + **GeeksforGeeks C++ Tutorials**: GeeksforGeeks C++
  + **w3schools C++ Tutorial**: w3schools C++
  + **StackOverflow**: If you face specific coding issues, StackOverflow can be a helpful resource for troubleshooting.

### 4. **Sample C++ Projects**:

* You might refer to other open-source grade calculator projects or examples of simple student grading systems to understand how they implement the grade calculation logic.
  + Example repositories:
    - GitHub repositories that host grade calculators or student management systems.

### 5. **Grading Systems and Criteria**:

* If your grading calculator is based on specific course or academic institution criteria, it may be useful to reference grading rubrics or guidelines from the educational institution or external grading systems.
  + Example: University course syllabus or educational standards documents.

**Code of The Project:**

#include <iostream>

#include <string>

#include <ctime>

#include <iomanip>

using namespace std;

struct Course {

string name;

int credits;

double quizMarks;

double assignmentMarks;

double midtermMarks;

double finalMarks;

double finalGrade;

};

double calculateFinalGrade(double quizMarks, double assignmentMarks, double midtermMarks, double finalMarks) {

return (quizMarks ) + (assignmentMarks ) + (midtermMarks ) + (finalMarks );

}

double calculateGPA(Course courses[], int numCourses) {

double totalMarks = 0;

int totalCredits = 0;

for (int i = 0; i < numCourses; i++) {

totalCredits += courses[i].credits;

totalMarks += courses[i].finalGrade \* courses[i].credits;

}

return totalMarks / totalCredits;

}

// Function to calculate grade based on final grade

char calculateGrade(double finalGrade) {

if (finalGrade >= 90) {

return 'A';

} else if (finalGrade >= 80) {

return 'B';

} else if (finalGrade >= 70) {

return 'C';

} else if (finalGrade >= 60) {

return 'D';

} else {

return 'F';

}

}

void displayCurrentDate() {

time\_t now = time(0);

tm\* localtm = localtime(&now);

cout << "\n\t\t\t\t\t\t\t\t\t==========================================" << endl;

cout << "\n\t\t\t\t\t\t\t\t\t\t Grade Based Calculator\n";

cout << "\n\t\t\t\t\t\t\t\t\t\t Date: "

<< 1900 + localtm->tm\_year << "-"

<< 1 + localtm->tm\_mon << "-"

<< localtm->tm\_mday << endl;

cout << "\n\t\t\t\t\t\t\t\t\t==========================================" << endl;

}

void displayEncouragingMessage(double gpa) {

if (gpa >= 3.7) {

cout << "\n\t\t\t\t\t Excellent work! Keep it up!" << endl;

} else if (gpa >= 3.0) {

cout << "\n\t\t\t\t\t Good job! You're doing well!" << endl;

} else if (gpa >= 2.0) {

cout << "\n\t\t\t\t\t Needs improvement. Keep pushing!" << endl;

} else {

cout << "\n\t\t\t\t\t Don't give up, there's always room for improvement!" << endl;

}

}

int main() {

string studentName;

displayCurrentDate();

cout << "\n\nEnter the student's name: ";

getline(cin, studentName);

int numCourses;

cout << "\nEnter the number of courses: ";

cin >> numCourses;

Course courses[numCourses];

for (int i = 0; i < numCourses; i++) {

cout << "\nEnter course " << i + 1 << " details:\n";

cout << "Course name: ";

cin.ignore(); // To clear the input buffer

getline(cin, courses[i].name);

cout << "Credits: ";

cin >> courses[i].credits;

while (courses[i].credits <= 0) {

cout << "Credits must be a positive number. Please enter again: ";

cin >> courses[i].credits;

}

cout << "Enter quiz marks (out of 20): ";

cin >> courses[i].quizMarks;

while (courses[i].quizMarks < 0 || courses[i].quizMarks > 20) {

cout << "Quiz marks must be between 0 and 20. Please enter again: ";

cin >> courses[i].quizMarks;

}

cout << "\nEnter assignment marks (out of 20): ";

cin >> courses[i].assignmentMarks;

while (courses[i].assignmentMarks < 0 || courses[i].assignmentMarks > 20) {

cout << "Assignment marks must be between 0 and 20. Please enter again: ";

cin >> courses[i].assignmentMarks;

}

cout << "\nEnter midterm marks (out of 20): ";

cin >> courses[i].midtermMarks;

while (courses[i].midtermMarks < 0 || courses[i].midtermMarks > 20) {

cout << "Midterm marks must be between 0 and 20. Please enter again: ";

cin >> courses[i].midtermMarks;

}

cout << "\nEnter final exam marks (out of 40): ";

cin >> courses[i].finalMarks;

while (courses[i].finalMarks < 0 || courses[i].finalMarks > 40) {

cout << "Final exam marks must be between 0 and 40. Please enter again: ";

cin >> courses[i].finalMarks;

}

courses[i].finalGrade = calculateFinalGrade(courses[i].quizMarks, courses[i].assignmentMarks, courses[i].midtermMarks, courses[i].finalMarks);

}

double gpa = calculateGPA(courses, numCourses);

cout << "\nCourse Grades for " << studentName << ":\n";

cout << setw(30) << left << "Course Name" << setw(15) << "Credits" << setw(15) << "Quiz" << setw(15) << "Assignment"

<< setw(15) << "Midterm" << setw(15) << "Final" << "Grade\n";

cout << string(100, '-') << endl;

for (int i = 0; i < numCourses; i++) {

cout << setw(30) << left << courses[i].name

<< setw(15) << courses[i].credits

<< setw(15) << courses[i].quizMarks

<< setw(15) << courses[i].assignmentMarks

<< setw(15) << courses[i].midtermMarks

<< setw(15) << courses[i].finalMarks

<< calculateGrade(courses[i].finalGrade) << endl;

}

cout << "\n" << studentName << ", your GPA is: " << fixed << setprecision(2) << gpa << endl;

displayEncouragingMessage(gpa);

return 0;

}